WHAT IS CLAIMED IS:

1. A system, comprising:

a processor; and

a memory comprising program instructions, wherein the program instructions are executable by the processor to implement a remote class loader mechanism configured to:

determine that a class is needed to execute code on the system;

obtain the class from a remote system via a network; and

store the class in a location indicated by a class path of a default class loader on the system.

2. The system as recited in claim 1, wherein the default class loader is configured to:

locate the class stored in the location indicated by the class path; and

load the class from the location for access by the code.

3. The system as recited in claim 1, wherein the location is a default directory for storing remote classes.

4. The system as recited in claim 1, wherein the location is a user-specified directory for storing remote classes.

5. The system as recited in claim 1, wherein, to determine that a class is needed to execute code on the system, the remote class loader mechanism is further configured to detect an exception generated by the code and indicating that the class is not on the system.

6. The system as recited in claim 1, wherein, to obtain the class from a remote system, the remote class loader mechanism is further configured to send a message requesting the class to one or more remote systems, wherein the message comprises information about the class for identifying a class file on the remote system that comprises the requested class.

7. The system as recited in claim 1, wherein, to obtain the class from a remote system, the remote class loader mechanism is further configured to:

send a message requesting the class to the remote system; and

receive the class from the remote system in one or more messages in response to the message.

8. The system as recited in claim 1, wherein, to obtain the class from a remote system, the remote class loader mechanism is further configured to:

broadcast a message requesting the class to one or more remote systems including the remote system on the network; and

receive the class from the remote system in one or more messages in response to the broadcast message.

9. The distributed computing system as recited in claim 8, wherein the one or more remote systems and the system are member peers of a peer group in a peer-to-peer network environment.

10. The system as recited in claim 1, wherein the program instructions are further executable by the processor to implement a virtual machine on the system, wherein the code is executable within the virtual machine.

11. The system as recited in claim 10, wherein the virtual machine is a Java Virtual Machine (JVM).

12. The system as recited in claim 1, wherein the code is in a bytecode computer language.

13. The system as recited in claim 1, wherein the code is Java code.

14. The system as recited in claim 1, wherein the system and the remote system are peer nodes configured to participate in a peer-to-peer environment on the network.

15. The system as recited in claim 1, wherein the system and the remote system are configured to participate as peer nodes in a peer-to-peer environment on the network in accordance with one or more peer-to-peer platform protocols for enabling the peer nodes to discover each other, communicate with each other, and cooperate with each other to form peer groups in the peer-to-peer environment.

16. The system as recited in claim 1, wherein the code is a code fragment of an application configured for execution on the system, and wherein the remote system is a node in a distributed computing framework that comprises the application and is configured to provide computer-executable code fragments of the application to two or more other systems to run the code fragments in parallel to execute the application.

17. The system as recited in claim 1, wherein the system and the remote system are configured to participate in a distributed computing system on the network for submitting computational tasks in a distributed heterogeneous networked environment that utilizes peer groups to decentralize task dispatching and post-processing functions and enables a plurality of jobs to be managed and run simultaneously.

18. A distributed computing system, comprising:

a master node configured to provide computer-executable code fragments of an application to a plurality of worker nodes on a network, wherein the code fragments are configured to run tasks in parallel on two or more of the plurality of worker nodes to perform a job;

a worker node configured to:

receive a code fragment from the master peer node;

determine that a class is needed to execute the code fragment;

obtain the class from a remote node via the network; and

store the class in a location indicated by a class path of a default class loader on the worker node.

19. The distributed computing system as recited in claim 18, wherein the default class loader is configured to:

locate the class stored in the location indicated by the class path; and

load the class from the location for access by the code fragment.

20. The distributed computing system as recited in claim 18, wherein the location is a default directory for storing remote classes.

21. The distributed computing system as recited in claim 18, wherein the location is a user-specified directory for storing remote classes.

22. The distributed computing system as recited in claim 18, wherein, to determine that a class is needed to execute the code fragment, the worker node is further configured to detect an exception generated by the code fragment and indicating that the class is not on the worker node.

23. The distributed computing system as recited in claim 18, wherein, to obtain the class from a remote node, the worker node is further configured to send a message requesting the class to the remote node, wherein the message comprises information about the class for identifying a class file that comprises the requested class.

24. The distributed computing system as recited in claim 18, wherein the master node is the remote node, and wherein, to obtain the class from a remote node, the worker node is further configured to:

send a message requesting the class to the master node; and

receive the class from the master node in one or more messages in response to the message.

25. The distributed computing system as recited in claim 18, wherein, to obtain the class from a remote node, the worker node is further configured to:

broadcast a message requesting the class to one or more remote nodes on the network; and

receive the class from the remote node in one or more messages in response to the broadcast message.

26. The distributed computing system as recited in claim 25 wherein the one or more remote nodes, the worker node, and the master nodes are member peers of a peer group in a peer-to-peer network environment.

27. The distributed computing system as recited in claim 25, wherein the one or more remote nodes are worker nodes configured to receive code fragments from the master node.

28. The distributed computing system as recited in claim 18, wherein the program instructions are further executable by the processor to implement a virtual machine on the worker node, wherein the code fragment is executable within the virtual machine.

29. The distributed computing system as recited in claim 28, wherein the virtual machine is a Java Virtual Machine (JVM).

30. The distributed computing system as recited in claim 18, wherein the code fragment is in a bytecode computer language.

31. The distributed computing system as recited in claim 18, wherein the code fragment is Java code.

32. The distributed computing system as recited in claim 18, wherein the worker node and the master node are peer nodes configured to participate in a peer-to-peer environment on the network.

33. The distributed computing system as recited in claim 18, wherein the worker node and the master node are configured to participate as peer nodes in a peer-to-peer environment on the network in accordance with one or more peer-to-peer platform protocols for enabling the peer nodes to discover each other, communicate with each other, and cooperate with each other to form peer groups in the peer-to-peer environment.

34. A system, comprising:

means for determining that a class is needed to execute code on a system;

means for obtaining the class from a remote system via a network; and

means for storing the class in a location indicated by a class path of a default class loader on the system.

35. A method, comprising:

determining that a class is needed to execute code on a system;

obtaining the class from a remote system via a network; and

storing the class in a location indicated by a class path of a default class loader on the system.

36. The method as recited in claim 35, further comprising:

the default class loader locating the class stored in the location indicated by the class path; and

the default class loader loading the class from the location for access by the code.

37. The method as recited in claim 35, wherein the location is a default directory for storing remote classes.

38. The method as recited in claim 35, wherein the location is a user-specified directory for storing remote classes.

39. The method as recited in claim 35, wherein said determining that a class is needed to execute code on the system comprises detecting an exception generated by the code and indicating that the class is not on the system.

40. The method as recited in claim 35, wherein obtaining the class from a remote system comprises sending a message requesting the class to one or more remote systems, wherein the message comprises information about the class for identifying a class file on the remote system that comprises the requested class.

41. The method as recited in claim 35, wherein said obtaining the class from a remote system comprises:

sending a message requesting the class to the remote system; and

receiving the class from the remote system in one or more messages in response to the message.

42. The method as recited in claim 35, wherein said obtaining the class from a remote system comprises:

broadcasting a message requesting the class to one or more remote systems on the network; and

receiving the class from the remote system in one or more messages in response to the broadcast message.

43. The method as recited in claim 42, wherein the one or more remote systems and the system are member peers of a peer group in a peer-to-peer network environment.

44. The method as recited in claim 35, wherein the code is executing within a virtual machine on the system.

45. The method as recited in claim 44, wherein the virtual machine is a Java Virtual Machine (JVM).

46. The method as recited in claim 35, wherein the code is in a bytecode computer language.

47. The method as recited in claim 35, wherein the code is Java code.

48. The method as recited in claim 35, wherein the system and the remote system are peer nodes configured to participate in a peer-to-peer environment on the network.

49. The method as recited in claim 35, wherein the system and the remote system are configured to participate as peer nodes in a peer-to-peer environment on the network in accordance with one or more peer-to-peer platform protocols for enabling the peer nodes

to discover each other, communicate with each other, and cooperate with each other to form peer groups in the peer-to-peer environment.

50. The method as recited in claim 35, wherein the code is a code fragment of an application configured for execution on the system, and wherein the remote system is a node in a distributed computing framework that comprises the application and is configured to provide computer-executable code fragments of the application to two or more other systems to run the code fragments in parallel to execute the application.

51. The method as recited in claim 35, wherein the system and the remote system are configured to participate in a distributed computing system on the network for submitting computational tasks in a distributed heterogeneous networked environment that utilizes peer groups to decentralize task dispatching and post-processing functions and enables a plurality of jobs to be managed and run simultaneously.

52. A computer-accessible medium comprising program instructions, wherein the program instructions are configured to implement:

determining that a class is needed to execute code on a system;

obtaining the class from a remote system via a network; and

storing the class in a location indicated by a class path of a default class loader on the system.

53. The computer-accessible medium as recited in claim 52, wherein the program instructions are further configured to implement:

the default class loader locating the class stored in the location indicated by the class path; and

the default class loader loading the class from the location for access by the code.

54. The computer-accessible medium as recited in claim 52, wherein the location is a default directory for storing remote classes.

55. The computer-accessible medium as recited in claim 52, wherein the location is a user-specified directory for storing remote classes.

56. The computer-accessible medium as recited in claim 52, wherein, in said determining that a class is needed to execute code on the system, the program instructions are further configured to implement detecting an exception generated by the code and indicating that the class is not on the system.

57. The computer-accessible medium as recited in claim 52, wherein, in said obtaining the class from a remote system, the program instructions are further configured to implement sending a message requesting the class to one or more remote systems, wherein the message comprises information about the class for identifying a class file on the remote system that comprises the requested class.

58. The computer-accessible medium as recited in claim 52, wherein, in said obtaining the class from a remote system, the program instructions are further configured to implement:

sending a message requesting the class to the remote system; and

receiving the class from the remote system in one or more messages in response to the message.

59. The computer-accessible medium as recited in claim 52, wherein, in said obtaining the class from a remote system, the program instructions are further configured to implement:

broadcasting a message requesting the class to one or more remote systems on the network; and

receiving the class from the remote system in one or more messages in response to the broadcast message.

60. The computer-accessible medium as recited in claim 59, wherein the one or more remote systems and the system are member peers of a peer group in a peer-to-peer network environment.

61. The computer-accessible medium as recited in claim 52, wherein the code is executing within a virtual machine on the system.

62. The computer-accessible medium as recited in claim 61, wherein the virtual machine is a Java Virtual Machine (JVM).

63. The computer-accessible medium as recited in claim 52, wherein the code is in a bytecode computer language.

64. The computer-accessible medium as recited in claim 52, wherein the code is Java code.

65. The computer-accessible medium as recited in claim 52, wherein the system and the remote system are peer nodes configured to participate in a peer-to-peer environment on the network.

66. The computer-accessible medium as recited in claim 52, wherein the system and the remote system are configured to participate as peer nodes in a peer-to-peer environment on the network in accordance with one or more peer-to-peer platform protocols for enabling the peer nodes to discover each other, communicate with each other, and cooperate with each other to form peer groups in the peer-to-peer environment.

67. The computer-accessible medium as recited in claim 52, wherein the code is a code fragment of an application configured for execution on the system, and wherein the remote system is a node in a distributed computing framework that comprises the application and is configured to provide computer-executable code fragments of the application to two or more other systems to run the code fragments in parallel to execute the application.

68. The computer-accessible medium as recited in claim 52, wherein the system and the remote system are configured to participate in a distributed computing system on the network for submitting computational tasks in a distributed heterogeneous networked environment that utilizes peer groups to decentralize task dispatching and post-processing functions and enables a plurality of jobs to be managed and run simultaneously.